Problem I Input file: i.in Output: monitor/stdout/cout/System.out

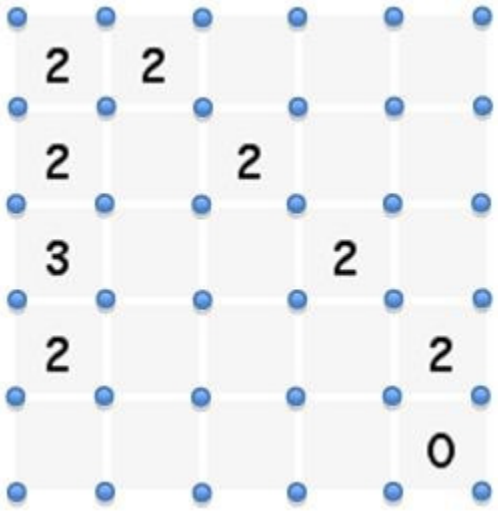
Containing Zombies and Respecting Building Codes

It's well documented that the only way to thwart zombies is to trap them in closed fences. Zombie hunters have for years been luring zombies into clear spaces and then erecting such fences at lightning speeds so they can't go very far.

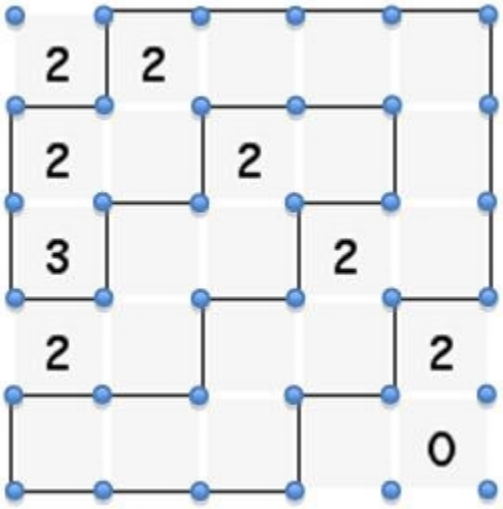
Unfortunately, government has its zoning laws, its taxes, and its building codes, and recently they started to enforce silly constraints on how these zombie-enclosing fences can be constructed. According to city codes, fence posts must be planted at regular intervals, and the fences themselves must respect arbitrary limits on how many walls can surround a unit square of land. We also want to build as long of an enclosing fence as possible, because the government has plenty of money to waste on such things.

Building Zombie Fences

Build walls between vertices to form a single enclosed fence without crossings or branches. The number indicates **exactly** how many walls — according to the crazy city building laws — must surround it (and a lack of number means there's no constraint.) So, presented with the following 5 x 5 grid of land squares:



the following fence could be constructed:



Problem I Input file: i.in Output: monitor/stdout/cout/System.out

The grid of lots is always `n x n` [$1 \leq n \leq 6$], and each lot is either a number (`0`, `1`, `2`, or `3`) to impose a constraint, or a blank if no constraint is being imposed. You're to output the length of the longest possible loop (or equivalently, the number of vertices in the loop), or `-1` if no loop exists. Note that loops of length 0 are invalid. A valid loop must enclose a non-zero amount of area.

Input

The input is a series of zombie fencing problems, expressed by `n`, the dimension of the problem, followed by an `n x n` grid ($1 \leq n \leq 6$) with the number constraints (with the - to represent no constraint), followed by a blank line. End of input is marked by a single `0`.

Output

For each fence problem, you should print the length of the longest fence loop that can be constructed for that problem while still respecting all constraints, or you should print `-1` if the problem has no such solution. There should be no extraneous white space, save for the newlines that separate each of the fence lengths.

Sample Input	**Sample Output**
2 22 22 3 222 222 222 5 ----0 2---2 3--2- 2-2-- 22--- 6 222222 2-22-2 22222- 22-2-2 2-22-2 222222 0	8 -1 32 46